

- anyway, the **exponential blowup** in the computation procedure is unavoidable
- let SAT be the problem of deciding if a formula in  $\mathcal{L}$  is satisfiable
- **Theorem 5. [Cook 1971]** *SAT is NP-complete.*

**Proof.** Translates every computation in a non-deterministic Turing machine into a propositional formula  $\square$

- the calculi that will be introduced later show several possible ways to implement a procedure for SAT
- they work on restricted set of formulas, so **normal forms** should be introduced

## Propositional Logic - normal forms

---

- two formulas  $F$  and  $G$  are (semantically) equivalent iff  $F \models G$  and  $G \models F$
- it is written  $F \equiv G$
- **Exercise:** show that this is equivalent to saying that  $F \Leftrightarrow G$  is a tautology
- let's review some named equivalences:

## Propositional Logic - normal forms

---

$(F \wedge F)$	$\equiv$	$F$	$\wedge$ -idempotency
$(F \vee F)$	$\equiv$	$F$	$\vee$ -idempotency
$(F \wedge G)$	$\equiv$	$(G \wedge F)$	$\wedge$ -conmutativity
$(F \vee G)$	$\equiv$	$(G \vee F)$	$\vee$ -conmutativity
$(F \wedge (G \wedge H))$	$\equiv$	$((F \wedge G) \wedge H)$	$\wedge$ -associativity
$(F \vee (G \vee H))$	$\equiv$	$((F \vee G) \vee H)$	$\vee$ -associativity
$(F \wedge G) \vee F$	$\equiv$	$F$	absorption
$(F \vee G) \wedge F$	$\equiv$	$F$	absorption
$(F \wedge (G \vee H))$	$\equiv$	$((F \wedge G) \vee (F \wedge H))$	distributivity
$(F \vee (G \wedge H))$	$\equiv$	$((F \vee G) \wedge (F \vee H))$	distributivity
$\neg\neg F$	$\equiv$	$F$	double negation
$\neg(F \wedge G)$	$\equiv$	$(\neg F \vee \neg G)$	de Morgan law
$\neg(F \vee G)$	$\equiv$	$(\neg F \wedge \neg G)$	de Morgan law
$F \leftrightarrow G$	$\equiv$	$(F \Rightarrow G) \wedge (G \Rightarrow F)$	equivalence
$F \Rightarrow G$	$\equiv$	$(\neg F \vee G)$	implication

- if we replace a subformula  $G$  of  $F$  by a formula  $H$  then we obtain  $F[G \setminus H]$
- **Theorem 6.** *If  $G$  is a subformula of  $F$ , and  $G \equiv H$  then  $F \equiv F[G \setminus H]$*
- **Exercise:** prove this theorem
- following the given equivalences, not all connectives are needed
- a set of connectives is said to be **necessary** iff any propositional formula can be transformed into a semantically equivalent one which only contains connectives in the set
- example:  $\{\neg, \wedge\}$  is a necessary set of connectives
- **Exercise:** prove this

- theorem 6 and the given equivalences can be used to introduce so-called **normal forms**
- a formula is in **negation normal form** iff it is built only by literals, conjunctions and disjunctions
- a formula is in **conjunctive normal form** (CNF) iff it has the form  $C_1 \wedge C_2 \wedge \dots \wedge C_n$  where each  $C_i$  is a disjunction of literals
- a formula is in **disjunctive normal form** (DNF) iff it has the form  $D_1 \vee D_2 \vee \dots \vee D_n$  where each  $D_i$  is a conjunction of literals
- $C_1 \wedge C_2 \wedge \dots \wedge C_n \equiv C_1 \wedge C_2 \wedge \dots \wedge C_n \wedge \top$ , so we can say  $\top$  is in CNF taking  $n = 0$
- $D_1 \vee D_2 \vee \dots \vee D_n \equiv D_1 \vee D_2 \vee \dots \vee D_n \vee \perp$ , so we can say  $\perp$  is in DNF taking  $n = 0$
- $C_i$  are called **clauses**;  $D_i$  are **dual clauses**; set notation is generally used
- **factoring** in clauses consists of eliminating duplicates of predicates in the same clause
- it is equivalent to applying the idempotency laws, although it is implied in set notation

Problem: transform a propositional formula into CNF

Input a propositional formula  $F$

Output a propositional formula  $G$  in CNF which is equivalent to  $F$

- 1: Eliminate all equivalence connectives using the equivalence law
- 2: Eliminate all implication connectives using the implication law
- 3: Eliminate all negation connectives (except those in front of propositions) using the de Morgan laws and the double negation law
- 4: Distribute all disjunctions over conjunctions using the distributivity, commutativity and associativity laws.

- example:  $(D \vee \neg A \vee B) \wedge (D \vee \neg A \vee \neg C)$  is the CNF of  $(A \wedge (B \Rightarrow C)) \Rightarrow D$

- this is a **correct** and **terminating** algorithm
- but the size of  $G$  can be exponential to the size of  $F$
- **Exercise:** show an example of this
- and also the structure of the formula is disrupted.
- **Exercise:** Write a Prolog program for computing the dual clause form of a propositional logic formula (the program for computing CNF will be given)
- Other transformations exist, even of polynomial time (see the definitional transformation technique in [Eder93]).

## Propositional Logic

---

- syntax
- semantics
- normal forms
- calculi
- algorithms
- propositional Horn logic

- having defined a logic, we're now interested in knowing whether the logical consequences can be mechanically computed
- a **calculus** consists in a set of **axioms** and a set of **inference rules** that intend to produce the logical consequences in a logic
- these elements define a **derivability relation** between a set of formulas  $\mathcal{F}$  and a formula  $G$

$$\mathcal{F} \vdash G$$

if  $G$  can be obtained from  $\mathcal{F}$  applying only inference rules and axioms.

- ideally, the derivability relation should be **sound** (ie if  $\mathcal{F} \vdash G$  then  $\mathcal{F} \models G$ )
- and also **complete** (ie if  $\mathcal{F} \models G$  then  $\mathcal{F} \vdash G$ )
- if a formula  $F$  can be derived in a theory  $\mathcal{F}$  using the axioms and inference rules of a calculus, then we say  $F$  is a **theorem** in  $\mathcal{F}$

- additionally, calculi can be classified as:
  - **negative** if its axioms are insatisfiable
  - **positive** if its axioms are valid
  - **generating** if theorems are derived from the axioms using the inference rules
  - **analyzing** if theorems are reduced to axioms using inference rules
- we will see some examples of different calculus in the following
  - natural deduction
  - sequent calculus
  - DPLL procedure
  - resolution
  - semantic tableaux

- **Natural Deduction** was introduced by Gentzen in 1935
- trying to formalize a logical reasoning in mathematics
- it is an example of a **positive** and **generating** calculus
- the idea is that when you're stating a proof, you make some assumptions, then draw conclusions, and finally, discharge the assumptions to obtain assumption-free results
- in consists of only one axiom  $\top$ , and several inference rules to produce proofs

Natural Deduction Inference Rules

constant rules

$$\frac{\perp}{F} \quad \frac{F \quad \neg F}{\perp}$$

negation introduction rule

$$\frac{\begin{array}{|c|} \hline F \\ \vdots \\ \perp \\ \hline \end{array}}{\neg F}$$

negation elimination rule

$$\frac{\begin{array}{|c|} \hline \neg F \\ \vdots \\ \perp \\ \hline \end{array}}{F}$$

Natural Deduction Inference Rules

and introduction rule

$$\frac{F \quad G}{F \wedge G}$$

and elimination rules

$$\frac{F \wedge G}{F} \quad \frac{F \wedge G}{G}$$

Natural Deduction Inference Rules

or introduction rules

$$\frac{\begin{array}{|c|} \hline \neg F \\ \vdots \\ G \\ \hline \end{array}}{F \vee G} \quad \frac{\begin{array}{|c|} \hline \neg G \\ \vdots \\ F \\ \hline \end{array}}{F \vee G}$$

or elimination rules

$$\frac{\begin{array}{c} \neg F \\ F \vee G \end{array}}{G} \quad \frac{\begin{array}{c} \neg G \\ F \vee G \end{array}}{F}$$

Natural Deduction Inference Rules

implication introduction rules

$$\frac{\begin{array}{|c|} \hline F \\ \vdots \\ G \\ \hline \end{array}}{F \Rightarrow G} \quad \frac{\begin{array}{|c|} \hline \neg G \\ \vdots \\ \neg F \\ \hline \end{array}}{F \Rightarrow G}$$

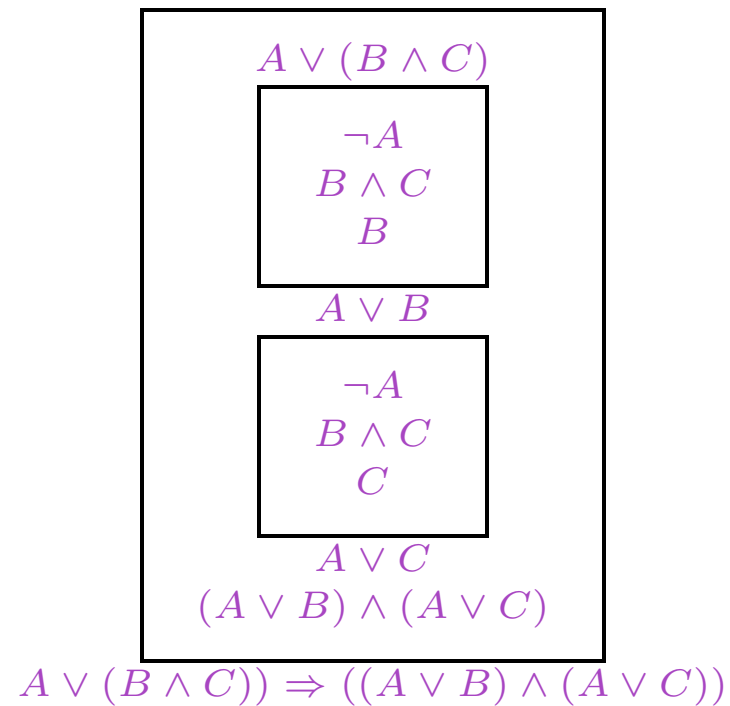
implication elimination rules

$$\frac{F \quad F \Rightarrow G}{G} \quad \frac{\neg G \quad F \Rightarrow G}{\neg F}$$

- the second constant rule is also called **contradiction**
- the negation introduction and elimination rules formalized the mathematical reasoning called **reduction ad absurdum**
- the first elimination rule for implication is also called **modus ponens**
- the second elimination rule is **modus tollens**
- the second introduction rule for implication is known as **contraposition**

- a **deduction** is a sequence of formulas, possibly included in open or closed boxes, such that each element is either
  - the axiom  $\top$
  - a formula that follows from earlier elements in the currently open boxes by applying rules of inferences  
(this may close boxes)
  - another formula, called **assumption**, which opens a new box
- a **proof** is a deduction in which all boxes are closed
- a **proof for  $F$**  is a proof in which  $F$  is the last formula in the deduction

Example of a proof for  $A \vee (B \wedge C) \Rightarrow ((A \vee B) \wedge (A \vee C))$ :



- repeated generation of the same deduction can be avoided in the form of [lemma](#)
- 

$$\frac{\neg\neg F}{\boxed{\begin{array}{c} \neg F \\ \perp \end{array}}} F$$

can be stated as a lemma  $\frac{\neg\neg F}{F}$

# Propositional Logic - calculi - natural deduction

---

- other useful lemmas

$$\begin{array}{c}
 \frac{\neg(F \wedge G) \quad F}{\neg G} \quad \frac{\neg(F \wedge G) \quad G}{\neg F} \quad \frac{\neg(F \vee G)}{(\neg F \wedge \neg G)} \\
 \\
 \begin{array}{c}
 \neg(F \wedge G) \\
 F \\
 \boxed{
 \begin{array}{c}
 G \\
 (F \wedge G) \\
 \perp
 \end{array}
 } \\
 \neg G
 \end{array}
 \quad
 \begin{array}{c}
 \neg(F \wedge G) \\
 G \\
 \boxed{
 \begin{array}{c}
 F \\
 (F \wedge G) \\
 \perp
 \end{array}
 } \\
 \neg G
 \end{array}
 \quad
 \begin{array}{c}
 \neg(F \vee G) \\
 \boxed{
 \begin{array}{c}
 \neg(\neg F \wedge \neg G) \\
 \boxed{
 \begin{array}{c}
 \neg G \\
 \neg\neg F \\
 F
 \end{array}
 } \\
 (F \vee G) \\
 \perp
 \end{array}
 } \\
 (\neg F \wedge \neg G)
 \end{array}
 \end{array}$$

- lemmas can help to shorten proofs
- but they can also enlarge the search space in the process of searching for proofs
- **Theorem 7.** *The propositional calculus of natural deduction is sound and complete*
- natural deduction calculus is easy to understand by humans, but difficult to generate by machines
- neither bottom-up nor top-down strategies are good in implementations

## Propositional Logic

---

- syntax
- semantics
- normal forms
- calculi
- algorithms
- propositional Horn logic

- to overcome the problems of natural deduction, Gentzen also introduced **sequent calculus**
- in sequent calculus, assumptions are replaced by **conditions**
- a **sequent** is an expression of the form  $\mathcal{F} \vdash \mathcal{G}$ , being  $\mathcal{F}, \mathcal{G}$  multisets of propositional formulas
- informally, it means that under conditions  $\mathcal{F}$  at least one of the formulas  $\mathcal{G}$  is provable
- inference rules are of the form

$$\frac{S_1 \dots S_n}{S} \mathbf{r}$$

where  $S_i, n \geq 0$  are sequents called **conditions**,  $S$  is a sequent called **conclusion**, and  $\mathbf{r}$  is the name of the rule

- in contrast to natural deduction, sequent calculus contains only introduction rules for each connective...
- but they have one of such rules **on the left**, and another **on the right** hand side of “ $\vdash$ ”
- there are also some **structural rules**

Sequent Calculus Axioms and Inference Rules

$$\frac{}{H, \mathcal{F} \vdash \mathcal{G}, H} \text{ axiom}$$

$$\frac{\mathcal{F} \vdash \mathcal{G}, H \quad H, \mathcal{F} \vdash \mathcal{G}}{\mathcal{F} \vdash \mathcal{G}} \text{ cut}$$

$$\frac{H, H, \mathcal{F} \vdash \mathcal{G}}{H, \mathcal{F} \vdash \mathcal{G}} \text{ lcontraction}$$

$$\frac{\mathcal{F} \vdash \mathcal{G}, H, H}{\mathcal{F} \vdash \mathcal{G}, H} \text{ rcontraction}$$

Sequent Calculus Axioms and Inference Rules

$$\frac{H_1, H_2, \mathcal{F} \vdash \mathcal{G}}{(H_1 \wedge H_2), \mathcal{F} \vdash \mathcal{G}} \wedge -\text{lintr.}$$

$$\frac{\mathcal{F} \vdash \mathcal{G}, H_1 \quad \mathcal{F} \vdash \mathcal{G}, H_2}{\mathcal{F} \vdash \mathcal{G}, (H_1 \wedge H_2)} \wedge -\text{rintr.}$$

$$\frac{H_1, \mathcal{F} \vdash \mathcal{G} \quad H_2, \mathcal{F} \vdash \mathcal{G}}{(H_1 \vee H_2), \mathcal{F} \vdash \mathcal{G}} \vee -\text{lintr.}$$

$$\frac{\mathcal{F} \vdash \mathcal{G}, H_1, H_2}{\mathcal{F} \vdash \mathcal{G}, (H_1 \vee H_2)} \vee -\text{rintr.}$$

Sequent Calculus Axioms and Inference Rules

$$\frac{\mathcal{F} \vdash \mathcal{G}, H}{\neg H, \mathcal{F} \vdash \mathcal{G}} \neg - \text{lintr.}$$

$$\frac{H, \mathcal{F} \vdash \mathcal{G}}{\mathcal{F} \vdash \mathcal{G}, \neg H} \neg - \text{rintr.}$$

$$\frac{\mathcal{F} \vdash \mathcal{G}, H_1 \quad H_2, \mathcal{F} \vdash \mathcal{G}}{(H_1 \Rightarrow H_2), \mathcal{F} \vdash \mathcal{G}} \Rightarrow - \text{lintr.}$$

$$\frac{H_1, \mathcal{F} \vdash \mathcal{G}, H_2}{\mathcal{F} \vdash \mathcal{G}, (H_1 \Rightarrow H_2)} \Rightarrow - \text{rintr.}$$

- a **proof** of a sequent  $S$  is inductively defined as:
  - an axiom  $S$  is a proof
  - if  $T_1, \dots, T_n$  are proofs of  $S_1, \dots, S_n$  respectively and there is an instance of an inference rule of the form

$$\frac{S_1, \dots, S_n}{S}$$

then the following is a proof

$$\frac{T_1, \dots, T_n}{S}$$

Examples of a proof in the sequent calculus

$$\frac{\frac{\overline{F \vdash F} \text{ axiom}}{\vdash F, \neg F} \neg\text{-rintr.}}{\vdash (F \vee \neg F)} \vee\text{-rintr.}$$

$$\frac{\frac{\overline{A, (C \Rightarrow D) \vdash B, A} \text{ axiom}}{\vdash A, (A \Rightarrow B), (C \Rightarrow D) \vdash B} \Rightarrow\text{-lintr.}}{\vdash A, (A \Rightarrow B), (C \Rightarrow D) \vdash B} \Rightarrow\text{-lintr.}$$

- **Exercise:** make a proof of the following sequents in sequent calculus:
  - $\vdash \neg A \vee A$
  - $\vdash ((A \Rightarrow B) \wedge (B \Rightarrow C)) \Rightarrow (A \Rightarrow C)$
  - $\vdash (A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$

- **Theorem 8.** *The propositional sequent calculus is sound and complete*
- if we view the sequent calculus as an analysing calculus, then cut is the only rule in which the condition contains a formula which does not occur in the conclusion
- from a proof search point of view, this is not desirable
- fortunately, Gentzen showed that applications of the cut rule can be eliminated
- but on the other hand, lemmas may shorten proof considerably
- intelligent and domain-dependant heuristics for using lemmas are necessary for guiding automatic theorem provers using sequent calculus

- the sequent calculus is the starting point of an area of research called **proof theory**
- it represents proofs as **formal mathematical objects**, facilitating their analysis by mathematical techniques
- as such, proof theory is closer to syntax, while model theory is more purely semantical